SEGMENTACIÓN DE OBJETOS

SHORT PROJECT

Edgar Perez Blanco

Bartomeu Perelló Comas

VC - GEI FIB

Cuatrimestre de Otoño 2019-2020

ÍNDICE

Planteamiento del problema

El problema planteado es crear un programa que, dada una imagen en la que el usuario selecciona un elemento que se distinga del fondo marcándolo con un rectángulo, este sea capaz de dibujar su contorno perfilado y localizarlo en otras imágenes o frames similares.

Para llevar a cabo esta tarea, a modo de resumen, la metodología es la siguiente:

1. Dividimos la imagen en un conjunto de sub imágenes o bloques.

2. Calculamos una serie de descriptores para cada uno de estos bloques.

3. Entrenamos un modelo de clasificación con los descriptores mencionados para distinguir entre dos clases (objeto - fondo). Los bloques que comprenden dentro de la marca del usuario se etiquetan como objeto, y los demás como fondo.

4. Volvemos a dividir la imagen en bloques, pero esta vez de menor tamaño para conseguir perfilar el contorno; calculamos los descriptores y solicitamos al modelo entrenado que nos diga si cada uno de los bloques es objeto o no.

Descriptores

FORMA: Histogram Oriented Gradients (HOG)

Para describir las los contornos de los objetos hemos creído conveniente utilizar los histogramas de orientación del gradiente (HOG). Este descriptores útil no sólo para diferenciar el contorno de un objeto sobre un fondo para marcarlo, sino también para realizar el tracking si el objeto cambia de posición.

Para cada muestra de entrenamiento/predicción proporcionada al clasificador se calculan 3 histogramas. Cuantos más histogramas se calculan, mayor es el tiempo de ejecución y el coste espacial del algoritmo. Para obtener un tiempo de procesamiento razonable y con buenos resultados establecimos esta cantidad.

Tambien notamos que aumentando el numero de histogramas el sistema tiende seguir los contornos fuertes (como por ejemplo las manchas de una vaca). Con número más alto de histogramas, el sistema marcaba solo las manchas negras como objeto, sin embargo, reduciendo la cantidad de estos, conseguimos que, aunque con menos precisión, el programa sea capaz de dibujar el contorno del animal incluyendo las zonas claras, patas, etc.

TEXTURA: Local Binary Patterns (LBP)

Tras hacer testing utilizando HOG e información sobre el color, notamos que en múltiples ocasiones el clasificador marcaba como objeto elementos de la imagen con un color parecido. Para poder diferenciar entre zonas no delimitadas por un contorno definido (en términos de color) decidimos incluir un descriptor de textura.

Para describir la textura decidimos utilizar los Local Binary Patterns ya que nos aporta una manera clara de diferenciar zonas de color similar. Estos han resultado ser muy útiles, y su inclusión mejoró notablemente los resultados del clasificador.

COLOR: HSV

Con el objetivo aportar más información de cada bloque, y teniendo en cuenta que las imágenes de entrada serán en color, hemos considerado conveniente la inclusión de información sobre el mismo.

En una primera iteración, decidimos incluir histogramas sobre el espacio de color HSV. Dividimos el histograma en rangos de valores, pero dado que utilizabamos bloques de información que pequeños, notamos que la gran mayoría de valores eran 0 y eran por lo tanto poco informativos. Nos topamos con un vector de características de más de 250 dimensiones solo para el HSV y solo se utilizaban en el mejor de los casos 60 de ellas. Por esta ineficiencia espacial y el coste temporal que conllevaba calcularlo nos decantamos por algo mucho más simple.

Simplemente decidimos calcular el valor medio para los tres valores del espacio HSV de cada uno de los bloques. De este modo, representamos de forma menos precisa, aunque a nuestro juicio suficientemente informativa, con tan solo 3 dimensiones, información sobre el color.

Clasificadores

Después de realizar pruebas con diferentes clasificadores, hemos puesto nuestro foco de atención en dos de ellos: la Support Vector Machine (SVM) y el TreeBagger.

SVM

Para objetos más homogéneos la utilización de una SVM es la que nos proporciona los mejores resultados. Si el objetivo del segmentador estuviese principalmente centrado en el trackeo de objetos en los frames posteriores, este a pesar de no marcar el objeto al completo (por ejemplo, de un ser humano segmenta la camiseta), el tracking lo realiza muy bien.

Con objetos relativamente lisos como un balón de baloncesto, tanto la segmentación como la localización del objeto da resultados muy positivos y muy precisos (aun si la selección inicial realizada por el usuario es altamente imprecisa.

También encontramos situaciones en las que el sistema no es capaz de obtener suficiente información para poder identificar al objeto y este falla. Aunque creemos que esto es debido a tener que seleccionar el objeto con un rectángulo. Si el objeto dentro del rectángulo, por culpa de su geometría, apenas cubre el 50% del área del mismo, el clasificador no es capaz de encontrar el objeto.

TREEBAGGER

A diferencia con las SVM, el clasificador TreeBagger es menos agresivo. De algún modo se ve más influenciado por la selección inicial, y por lo tanto, debido a la imprecisión de esta, el objeto suele estar contenido en su seleccion resultado, pero no de una forma tan precisa como conseguimos con una SVM.

Entonces para mejor fiabilidad de marcar la totalidad del objeto cuando no es tan homogéneo o que funciona prácticamente siempre con menos precisión hemos optado por el TreeBagger de 100 árboles.

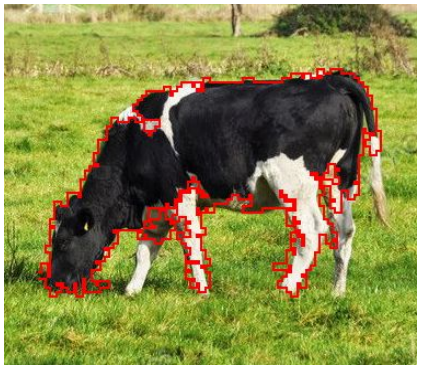
Este nos da más confianza de contener más pixeles de objeto en su selección final, a cambio de dar también más falsos positivos.

EJEMPLOS

A continuación vamos a intentar reforzar con un ejemplo la anterior explicación. Tomando la siguiente foto como entrada del programa obtenemos diferentes resultados en el marcado del objeto.

Imagen original con la marca del usuario:

Las siguientes imagenes corresponden con la salida de los dos clasificadores:



TREEBAGGER

SVM

En este caso el TreeBagger nos da un mejor resultado ya que la SVM tiende siempre a seleccionar un color si el objeto elegido está formado por varios. Pero la precisión de selección de la SVM es superior.

A continuación se muestran los datos obtenidos después de realizar la clasificación en ambos clasificadores, las tablas que se encuentran en la parte superior izquierda corresponden con el resultado de la matriz de confusión, la de la parte superior derecha indica la parte de la clase que se ha clasificado correctamente y la tabla de la parte inferior izquierda hace referencia a los falsos positivos de la ejecución.

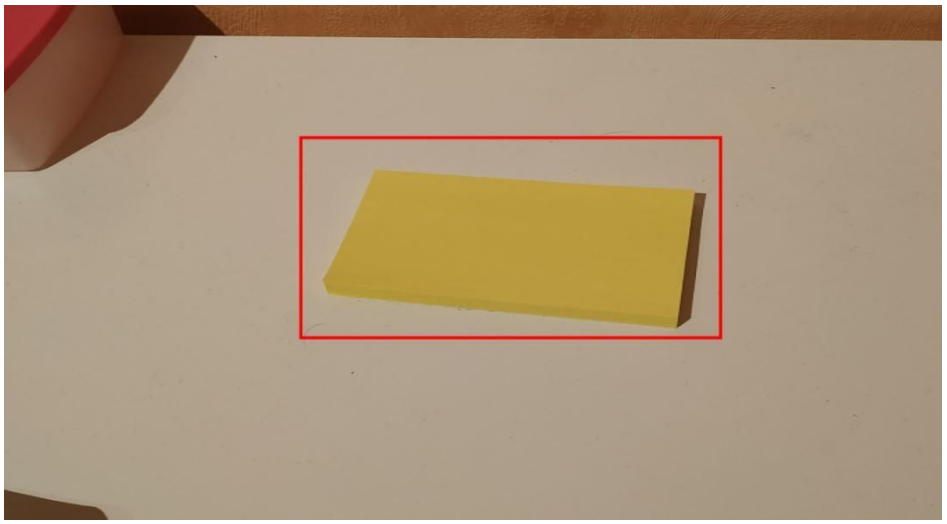
Ejecución con el clasificador **TreeBagger**:

El siguiente conjunto pertenece al clasificador **SVM**:

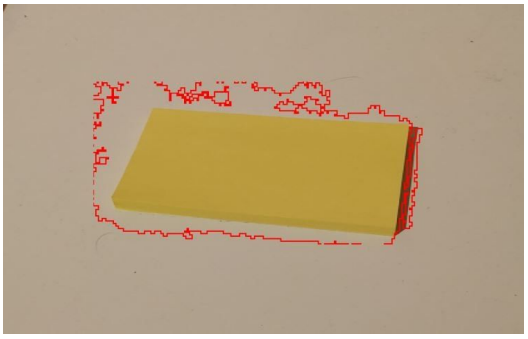
Como se puede apreciar en la tabla de falsos positivos el rendimiento del TreeBagger es bastante superior en este tipo de objetos ya que hay un 20% diferencia entre ellos. Cabe destacar que si nos referimos a la asignación de Objeto podemos comprobar que la SVM tiende a seleccionar solo pixeles que forman parte del objeto deseado aunque es este caso el TreeTagger tampoco se ha errado tanto.

En el siguiente ejemplo podremos apreciar las ventajas de la SVM frente al TreeBagger si el objeto es más homogéneo.

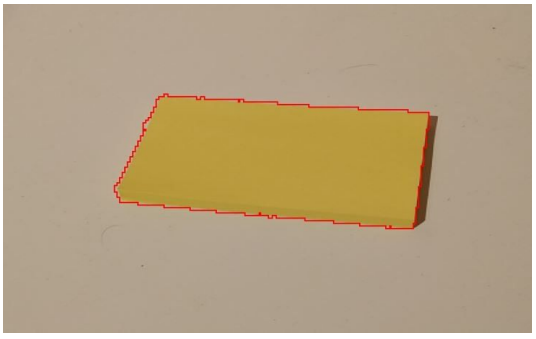
Imagen original con la marca del usuario:



Las siguientes imagenes corresponden con la salida de los dos clasificadores:



TREEBAGGER

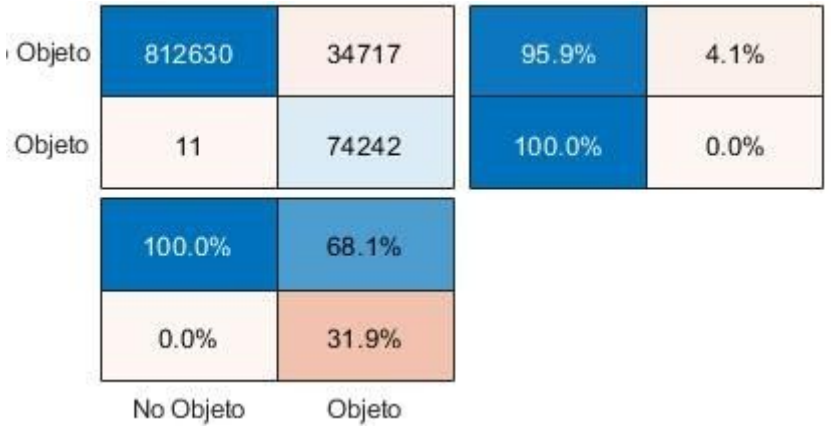


SVM

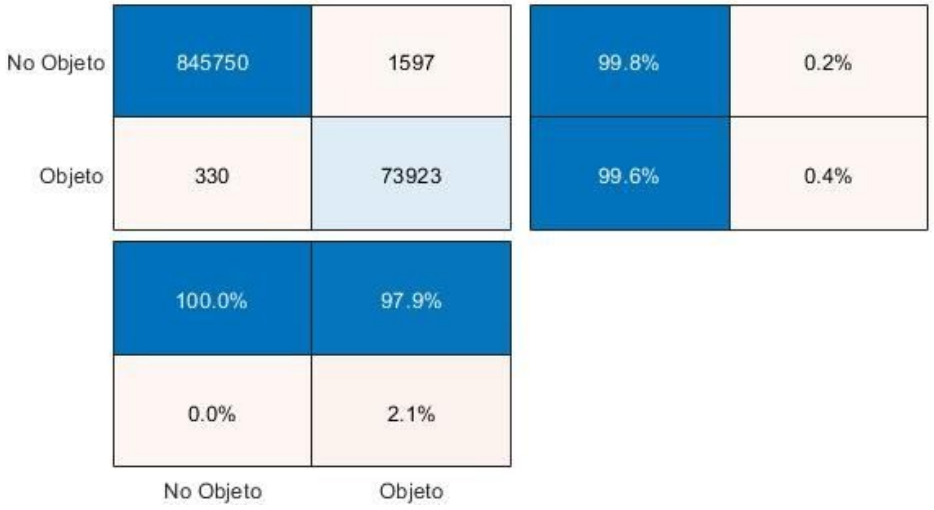
En esta situación deducimos que el TreeBagger se ve más influenciado por la textura de la imagen y como en ambos objetos las superficies son prácticamente lisas este opta por seleccionar parte de la mesa como objeto; ya que eso le hemos dicho a la hora de seleccionar el objeto en la imagen original.

Por otra parte la SVM es capaz de diferenciar correctamente por el color y segmentar de forma casi perfecta el objeto.

A continuación vamos a comparar los datos obtenidos en ambas ejecuciones:



TREEBAGGER



SVM

Se puede apreciar en la tabla de falsos positivos en ambos casos que al menos han podido detectar correctamente la totalidad del objeto pero si comprobamos la tabla que hace referencia al porcentaje de objeto seleccionado podemos ver que el TreeBagger no lo ha realizado de forma muy correcta ya que una tercera parte son falsos positivos, en cambio en la SVM solo un 2% son falsos positivos.

Conclusión

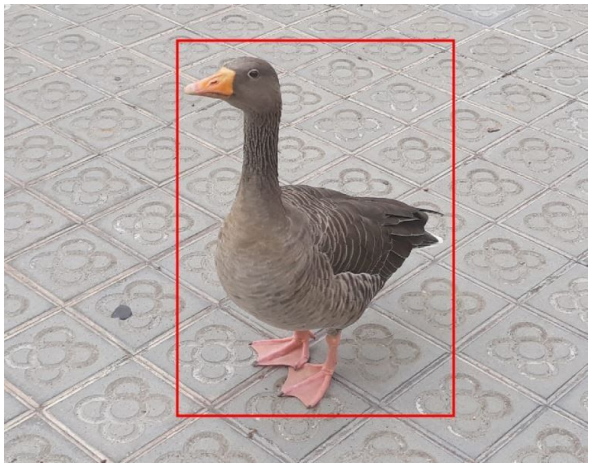
A modo de conclusión sobre los clasificadores, creemos que ambos son convenientes de mantener. En nuestro sistema damos la opción de escoger cual se quiere usar y dependiendo de la aplicación de este, o más bien, del tipo de objeto a reconocer, se debería escoger uno u otro.

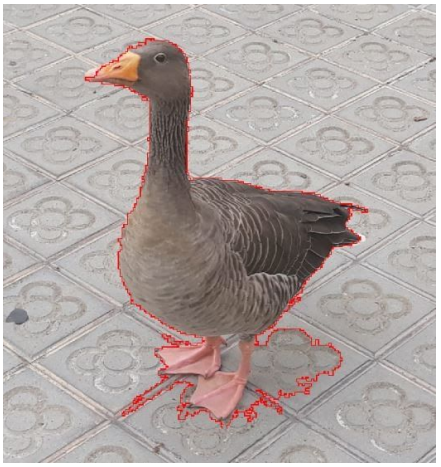
Desde nuestro punto de vista, para objetos de “fácil” segmentación o situaciones en las que no queremos falsos positivos en la clase objeto deberíamos usar la SVM, pero para objetos más “complejos” o donde no queremos falsos negativos en clase objeto, el TreeBagger.

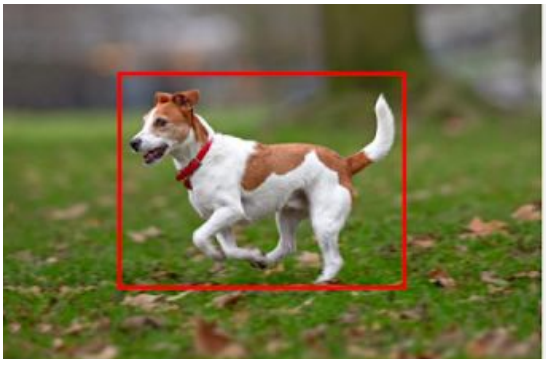
ANEXO

A continuación mostramos algunos ejemplos del funcionamiento del programa:

Imágenes:







Videos: